

### Integrantes:

- Camilo Molina Plata - 202221119
- Samuel Rozen Mogollon

## | 1. Identificación de componentes (Sección 3 - Análisis)

### | 1.1 ¿Cuáles son los **parámetros** de la función `generate_tone` ?

- **out (short \*)**: Es un puntero al arreglo donde se guardarán las muestras de audio generadas.
- **nSamples (int)**: Número total de muestras que se deben generar.
- **freqHz (unsigned int)**: La frecuencia del tono en hercios (Hz).
- **amp (short)**: La amplitud del tono (volumen) en formato de punto fijo Q15.

### | 1.2 ¿Cuáles son las **variables locales** de la función `generate_tone` ?

- **phase (unsigned int)**: Acumulador de fase, inicializado en 0.
- **phaseStep (unsigned int)**: El incremento de fase calculado.
- **temp (int)**: Variable temporal para almacenar el resultado de la multiplicación (en formato Q30).
- **sample (short)**: El valor final de la muestra después de regresar a formato Q15.
- **i (int)**: Contador para el ciclo for.
- **index (unsigned char)**: Índice de 8 bits para buscar valores en la tabla de senos (sineTable).

## | 2. Diagramas de la Pila (Sección 3 - Análisis)

### | 2.1 **Esquema 1**: Representación de la pila **justo antes** de la primera instrucción de la función (momento del prólogo).

Dirección	Contenido	Desplazamiento
Alta (Memoria)	amp (short, alineado a 32 bits)	[ebp + 20]
	freqHz (unsigned int)	[ebp + 16]
	nSamples (int)	[ebp + 12]
	out (short*)	[ebp + 8]
	Dirección de retorno (puesta por call)	[ebp + 4]
EBP / ESP →	EBP del llamador (Saved EBP)	[ebp + 0]
Baja (Memoria)	(Espacio disponible / Libre)	

### | 2.2 **Esquema 2**: Representación de la pila **justo antes del retorno** ( `ret` ), después de haber ejecutado la última instrucción.

Dirección	Contenido	Desplazamiento
Alta (Memoria)	amp (short, alineado a 32 bits)	[ebp + 20]
	freqHz (unsigned int)	[ebp + 16]
	nSamples (int)	[ebp + 12]

Dirección	Contenido	Desplazamiento
	out (short*)	[ebp + 8]
	Dirección de retorno	[ebp + 4]
EBP →	EBP del llamador (Saved EBP)	[ebp + 0]
	phase (unsigned int)	[ebp - 4]
	phaseStep (unsigned int)	[ebp - 8]
	i (int - contador del ciclo)	[ebp - 12]
	temp (int - resultado Q30)	[ebp - 16]
ESP →	index (unsigned char, alineado a 4 bytes)	[ebp - 20]
Baja (Memoria)		

## | 3. Justificación de los **desplazamientos** (offsets)

### | 3.1 Parametros

- **[ebp + 0] (0 bytes)**: Contiene el **EBP guardado** del llamador. Ocupa 4 bytes.
- **[ebp + 4] (4 bytes)**: Contiene la **Dirección de Retorno**. Esta dirección fue puesta por la instrucción call. Ocupa 4 bytes.
- **[ebp + 8] (8 bytes)**: Parámetro **out** (puntero, 4 bytes). Es el primer parámetro de la función (el más cercano al retorno).
- **[ebp + 12] (12 bytes)**: Parámetro **nSamples** (int, 4 bytes). Se encuentra 4 bytes después de out.
- **[ebp + 16] (16 bytes)**: Parámetro **freqHz** (unsigned int, 4 bytes). Se encuentra 4 bytes después de nSamples.
- **[ebp + 20] (20 bytes)**: Parámetro **amp** (short). Aunque un short mide 2 bytes, en la arquitectura de 32 bits los argumentos se "alinean" a **4 bytes** al ser empujados a la pila para mantener la eficiencia del procesador.

### | 3.2 Variables locales

- **[ebp - 4]**: Variable **phase** (unsigned int, 4 bytes). Es la primera variable declarada.
- **[ebp - 8]**: Variable **phaseStep** (unsigned int, 4 bytes)
- **[ebp - 12]**: Variable **temp** (int, 4 bytes)
- **[ebp - 16]**: Variable **i** (int, 4 bytes). Se usa como contador del for.
- **[ebp - 20]**: Variable **sample** (short). Se le asignan 4 bytes por alineación.
- **[ebp - 24]**: Variable **index** (unsigned char). Se le asignan 4 bytes por alineación

## | 4. Descripción de la pila del procedimiento traducido

Para la implementación de la función en lenguaje ensamblador, se utilizó el atributo `__declspec(naked)`. Esto implica que el compilador no genera automáticamente el prólogo ni el epílogo de la función, por lo cual la gestión de la pila (stack) se realizó de forma manual para garantizar el control total sobre los registros y la memoria.

### | 4.1. Gestión del Marco de Pila (Prólogo)

- **push ebp**: Guarda el puntero de base del llamador para poder restaurarlo al finalizar.
- **mov ebp, esp**: Establece el inicio de nuestro marco de pila actual. A partir de este momento, cualquier referencia a **[ebp + X]** accederá a los parámetros y **[ebp - X]** a las variables locales.
- **sub esp, 20**: Se reservaron **20 bytes** en la pila. Esto permite alojar las variables locales definidas en el código C (**phase**, **phaseStep**, **i**, **temp**, e **index**). Se asignaron 4 bytes a cada una para mantener la alineación de la pila.

## 4.2 Mapeo de Variables y Parámetros

En la implementación, el acceso a los datos se justificó mediante los siguientes desplazamientos (offsets):

Dato	Ubicación	Justificación
out	[ebp + 8]	Primer parámetro (puntero).
nSamples	[ebp + 12]	Segundo parámetro.
freqHz	[ebp + 16]	Tercer parámetro.
amp	[ebp + 20]	Cuarto parámetro (alineado a 4 bytes).
phase	[ebp - 4]	Variable local de 4 bytes.
phaseStep	[ebp - 8]	Variable local de 4 bytes.
i	[ebp - 12]	Contador del bucle.
temp	[ebp - 16]	Resultado intermedio (Q30).
index	[ebp - 20]	Índice para la tabla de senos.

*Nota: La variable `sample` no se guardó permanentemente en la pila, sino que se procesó directamente en el registro `eax` antes de escribirse en el arreglo `out`, con el fin de optimizar el uso de la memoria.*

## 4.3 Llamadas a Funciones Auxiliares

Para ejecutar `freq_to_phaseStep(freqHz)`, se siguió el protocolo de la arquitectura:

1. Se empujó el parámetro a la pila: `push dword ptr [ebp + 16]`.
2. Se realizó el `call`. El resultado, según la convención, fue retornado en el registro `eax`.
3. Se realizó la limpieza de la pila con `add esp, 4` para mantener la integridad del registro `esp`.

## 4.4 Finalización (Epílogo)

Para salir de la función de forma segura y devolver el control al programa principal:

- `mov esp, ebp`: Se descartan las variables locales moviendo el puntero de la pila de vuelta a la base del marco.
- `pop ebp`: Se restaura el valor original del `ebp` del llamador.
- `ret`: Se salta a la dirección de retorno almacenada en `[ebp + 4]`.